Software User Manual (SUM)

for the

ALERT SERVICES VERSION 1.3.4.2

26 March 1997

Sun Solaris 2.5.1

Program Manager
Common Hardware Software
Fort Monmouth, NJ 07703

Table of Contents

1. Scope

1.1. Identification

Common Operating Environment (COE) Alerts Software Version 1.3.4.2 on the following platforms:
SUN SPARC20 Solaris 2.5.1.

1.2. System Overview

The COE Alerts Software allows its users to send, receive, and manage alerts. These alerts consist of sender-specified text, a sender-specified classification, and a sender-specified priority, as well as sender identification, a time-tag, and a sender-specified alert-ID. The receiver of alerts can control what are received by means of an alert-ID mask, which is a UNIX regular expression which an incoming alert must match in order for the receiver to be notified of the alert. The sender also has the option of specifying that an alert is “persistent”. Persistent alerts are stored in a database when sent, and can survive a system shutdown.

1.3. Document Overview

This document is a hands-on software user manual. It’s purpose is to provide instructions on using the Alert software. The document is organized into the following sections:
1. Scope
2. Referenced Documents
3. Software Summary
4. Access to Software
5. Processing Reference Guide
6. Notes

2. Referenced Documents

SUN SPARC 20 Solaris 2.5.1 Operating System Documents

3. Software Summary

3.1. Software Application

The COE Alerts Software provides client applications with the capability to send, receive, and respond to alerts of varying degrees of urgency and classification level using a common software interface.

3.2. Software Inventory

The Alerts Software is provided in the form of executable program, source, and scripts. The following is a list of each directory and the files it contains:

- bin directory

    alerts_server

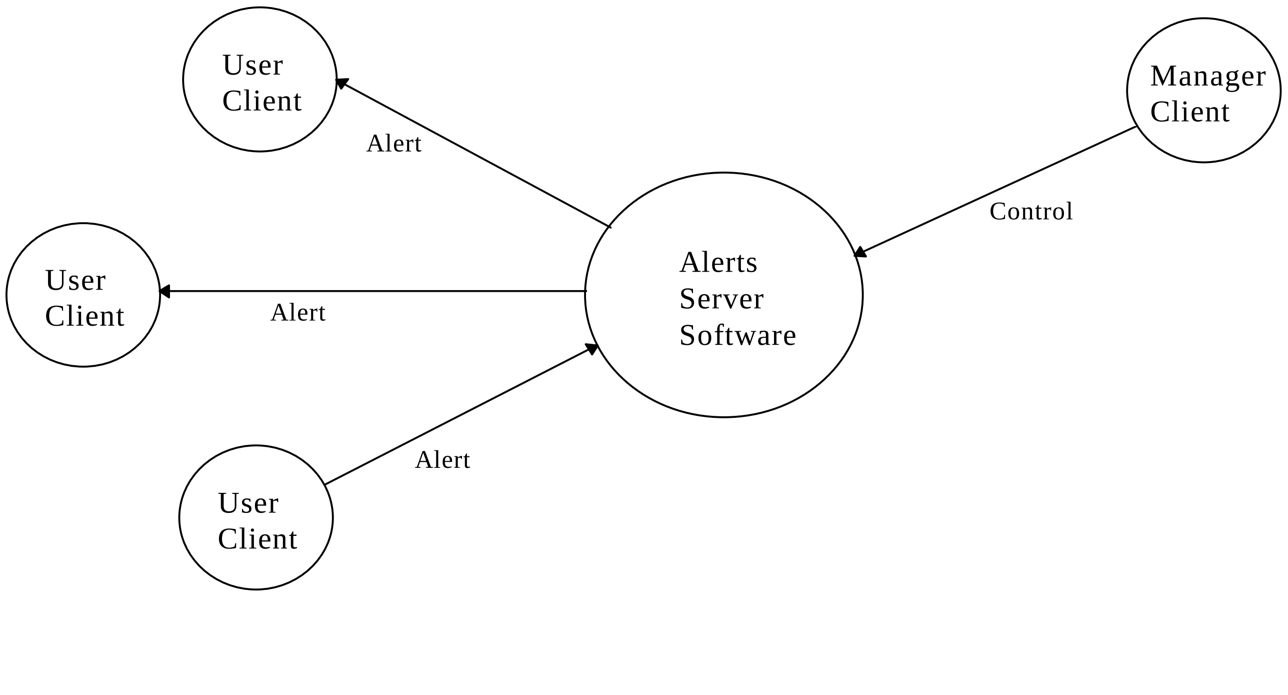
run_alerts
run_test_alert
run_test_display
run_ctest_alert
run_ctest_display
server_shutdown
test_alert_client
test_display_client
test_c_alert_client
test_c_display_client
up

3.3. Software Environment

The COE Alerts Software runs on networks of SUN SPARC20s running Solaris 2.5.1 under the Unix operating system.

3.4. Software Organization and Overview of Operation

The COE Alerts Software is organized into two distinct categories: Alert Server Software, and Alert Client Software. The diagram below shows the interaction between the Alert Server and Client software. Descriptions for these categories follow the diagram.



Alert server software

All Alert Server Software contributes only to the alerts_server process, which acts as a mediator between Alert client applications, passing alerts back and forth between them, storing "persistent" alerts in the database, and retrieving them when necessary.

Alert client software

Ada or C applications that wish to send or receive alerts gain access to the alerts_server process via the Alert Client Software. Such applications can be divided into two occasionally overlapping categories: Alert User clients, who only need to send and receive alerts and responses, and Alert Manager clients, who need to be able to control the way that the alerts_server process behaves. These two groups of clients use two distinct subgroups of the Alert Client Software.

Alert user clients

An Alert User Clients application consists of routines contained in the Alert_Display_Operations and Alert_Client_Operations packages or C programs. These routines provide the following capabilities: registration with the alerts_server process, filtering of incoming alerts, receiving and sending alerts.

Alert manager clients

An Alert Manager Clients application consists of routines contained in the Alert_Management_Operations package or C program. These routines provide the following capabilities: suspension and resumption of alert notification, disabling and enabling of processing of non-critical alerts by the alerts_server, and disabling and enabling the capability for suspension and resumption of alert notification.

3.5. Contingencies and Alternate States and Modes of Operation

The COE Alerts Software is embedded into a host system and adopts the modes and states of that system.

3.6. Security and Privacy

The COE Alerts Software has been implemented to support a "system-high" security regimen (i.e. it depends on its host system for security services and support.)

3.7. Assistance and Problem Reporting

Submit all Common Software (CS) problem reports to "The Common Software Bulletin Board".

4. Access to the Software

4.1. First-Time User of the Software

4.1.1. Equipment Familiarization

Alerts v1.3.4.2 runs on networks of SUN SPARC20s running Solaris 2.5.1 under the Unix operating system. Procedures for turning on and off power, keyboard layout ,and appearance of the cursor can be found in a SUN reference manual.

4.1.2. Access Control

Alerts v1.3.4.2 does not have any security features. Any passwords needed, such as user log in, should be provided.

4.1.3. Installation and Setup

Alerts v1.3.4.2 will be installable from an 8 mm tape in DII COE segmented format. Log onto the DII workstation (onto which the software is to be installed) as ‘sysadmin’. Insert the tape into the tape drive and start the COE Segment Installer Tool by selecting ‘Segment Installer’ from the ‘Software’ pulldown menu. Click on ‘Select Source’ in the Installer window and then select ‘Other’. In the input area, type ‘/dev/rmt/0mn’ (or the tape device file that’s appropriate for your system), and then click ‘Ok”. Click on the Read Contents button to read the contents of the tape. When the Installer is finished reading the tape, an entry for the Alerts segment will appear at the bottom of the Installer window. Select the Alerts segment (by clicking it once) and then click the Install button. Alerts v1.3.4.2 will be installed under the /h/ALRTSV directory. Once the software has been installed on a machine, a TCP/IP port number needs to be reserved for the server so that clients on either a LAN or WAN can attach to the server. To accomplish this, the alerts_server process requires a socket entry in the /etc/services file. Verify that the following line has been added to that file:

```
ALERTS      8001/tcp    alerts_server
```

4.2. Initiating a Session

Change directories to the directory where the Alerts software was installed. The alerts base directory for Version 1.3.4.2 is ALRTSV.

```
$ cd /h/ALRTSV
```

Only one alert_server process can run on a single machine at a time. Before the alerts_server process can be started, verify that there is not already another server running. Use the following command to check for an existing server :

```
$ ps -ef | grep ALERTS
```

If the output of this command shows an existing server, it needs to be shutdown before a new server can be started. This is accomplished with the server_shutdown command located in the /h/ALRTSV/bin directory.

```
$ cd /h/ALRTSV/bin
$ server_shutdown
```

The alerts_server process can then be started by running the following script:

```
$ run_alerts
```

Initially, the script adds /h/ALRTSV/bin to the PATH environment variable, sets the DISPLAY variable to the host name of the machine on which the script was started, and sets the ALERTS environment variable to /h/ALRTSV. Next, the script sets the ALERT_DB environment variable to the name of the alerts database. This environment variable is combined with the host name of the machine on which the server is running to create the file into which persistent alerts are stored. The server process will create this file if it does not already exist. The script also sets the ALERTS_HOST environment variable to the host name of the machine on which the script was started. Any clients that which to attach to the server must also have the ALERTS_HOST environment variable set to the host name of the machine on which the server is running. The alerts_server process uses UNIX sockets to perform inter-process communication, and cannot be initiated if there are any old connections waiting to time out. The run_alerts script checks to see if there are any connections still waiting to time out. Sockets can take as long as 5 minutes to shutdown, so the script will wait until they go away. When the process finally starts, the following message appears:

```
 Starting alerts_server ...
```

The alerts server should now be up and running.

4.3. Stopping and Suspending Work

Before the alerts_server process can be started, it is necessary to ensure that there is not another alerts_server process already running. A server_shutdown command should be executed first:

```
      $ bin/server_shutdown
```

5. Processing Reference Guide

5.1. Capabilities

The Alert Services v1.3.4.2 uses TCP/IP (Berkley sockets) to communicate between client and server. This architecture allows for many different configurations of clients and servers. The Alerts Services can be accessed from clients throughout a LAN or WAN environment. Client/Server sessions can be run in different configurations; such as having the server and client processes running on the same machine, or on different machines on the same LAN, or on different machines on separate LANS connected by a router. For an application to communicate with a server across the network, the alert server in the /etc/services file on each system must specify the same port number.

There are four possible configurations that the Alert Services can run in.

<u>Configuration One</u>

The server process and both client codes reside on one SPARC20 Configuration

<u>Configuration Two</u>

The server process resides on one SPARC20 and both client codes reside on different machines.

<u>Configuration Three</u>

Multiple server processes can exist on the LAN as long as there is no more than one server process per machine. Any client processes on the LAN can attach to a specific server process by setting the environment variable ALERTS_HOST to the host name of the machine running that server process.

Configuration four illustrates the use of Alerts over separate LANS using a router.

<u>Configuration Four</u>

Multiple server processes can exist on one LAN and any number of clients can exist on a separate LAN, with both LANs being connected via a router. As in Configuration Three, any client processes on the LAN can attach to a specific server process by setting the environment variable ALERTS_HOST to the host name of the machine running that server process.

5.2. Conventions

Not applicable. There are no conventions used by the software, such as the use of colors in displays, the use of audible alarms, the use of abbreviated vocabulary, and the use of rules for assigning names or codes.

5.3. Processing Procedures

5.3.x. [Aspect of Software Use]

There are no specific user-interfaces with the Alerts Server. Any user who will interface with the Alerts Server will do so via client software and API calls.

5.4. Related Processing

The server process performs all operations in the background with no need for user intervention. When the server is brought up, it first establishes a listen socket on which it waits for clients to attach. When a client wants to register for notification, create, or delete an alert, it must first receive a new socket on which to be serviced by the server. The server creates a task for that client and a new socket for it, and goes back to listening for others. As clients attach, tasks are created to handle their requests.

The alerts server maintains three internal queues: alert notification queue, alert response queue, and the alert queue. When applications register for alert notification or response, the alert ID pattern and application socket ID are placed in the corresponding queue. When an alert is generated, it is put in the alert queue. If the alert is marked as persistent, it is stored in a database or file. Once the alert is stored, the alert server looks through the notification queue to find applications whose pattern matches the alert ID. When a match is found, the alert is forwarded to the application. When an alert response is received, a similar process is invoked using the response queue.

5.5. Data Backup

Not applicable.

5.6. Recovery from Errors, Malfunctions, and Emergencies

If the server process goes down for some reason, any alert that was marked as persistent will remain in the alerts_db file and will be present when the server is brought back up. Any other alerts will be lost. If the server crashes, the run_alerts script needs to be re-started.

5.7. Messages

The following error messages can occur during the execution of the Alerts Server:

```
Alert server failed to read client request
```
Cause: The server has encountered an error while trying to receive data from the socket connection it has with the client.

```
Alert printer not defined, using default
```
Cause: The environment variable ALERTS_PRINTER has not been set. The variable takes on a default value and produces the error message.

```
Alerts server failed to establish listen socket
```
Cause: The server could not establish a socket on which to listen for new clients. Two reasons why a user will see this error message are:

1) A user tries to start a server on a machine already running a server
2) An old listen socket is waiting to time out.

NOTE
The run_alerts script will check for any old listen sockets and display a message that the socket is waiting to time out before starting the server process.

```
Alerts server failed to convert listen socket
```
Cause: The server could not convert listen socket to non-blocking. The socket needs to be non-blocking because if an IO request causes the server process to become blocked, NONE of its tasks can execute until the IO completes.

```
Alerts server failed to accept client
```

Cause: A new socket connection can not be made with the client requesting services.

```
Error in bind
```

Cause: After socket was created, the socket descriptor could not be bound to an address.
This can occur any time a socket is created, whether the socket is a server listen socket or a socket for an individual client.

5.8. Quick-Reference Guide

Not Applicable

6. Notes

COE	Common Operating Environment
CS	Common Software
TCP/IP	Transmission Control Protocol / Internet Protocol
LAN	Local Area Network